Kamailio World 2018

Dispatcher gateway monitoring and Load Balancing With Congestion Detection

Julien Chavanton

Lead software engineer - voice routing @ flowroute.com

01/24/2018

Presentation summary

- Latency monitoring and congestion estimation
- Benefits of the new algorithm
- Configuring load balancing with congestion detection
- Expected behavior with examples
- Laboratory A/B testing and results

Latency monitoring in Kamailio’s dispatcher module

Added in 2017

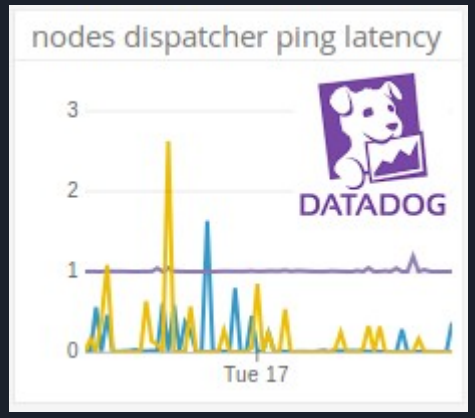
https://www.kamailio.org/w/2017/12/dispatcher-latency-stats-monitoring-with-statsd/

```
modparam("dispatcher", "ds_ping_interval", 1)                    // send SIP OPTIONS
modparam("dispatcher", "ds_ping_latency_stats", 1)               // ON/OFF

modparam("dispatcher", "ds_latency_estimator_alpha", 900)
// 900/1000 this is controlling the responsiveness and memory of the EWMA
```

kamcmd dispatcher.list

```
        URI: sip:14.56.98.51:5060
        FLAGS: AP
        PRIORITY: 12
        ATTRS: {
                BODY: weight=50;rweight=50
                DUID:
                MAXLOAD: 0
                WEIGHT: 50
                RWEIGHT: 50
                SOCKET:
        }
        LATENCY: {
                AVG: 72.750000
                STD: 0.500000
                EST: 98.750000  # 98ms – 72ms = +26ms (estimated congestion_ms)
                MAX: 112
                TIMEOUT: 0        # count of SIP OPTION timeouts (>fr_timer) default 30s
        }
```



New algorithm, why ?

- algorithm **objective** : Minimize the traffic sent to congested gateways
- **benefits** : minimize the impact on media and signaling resulting in improved connectivity and audio quality when gateways or networks are facing problems.
- current **alternative** :
  The best solution available, is to detect gateway timeout using SIP OPTION pings and automatically disable unresponsive gateways.

```
modparam("dispatcher", "ds_ping_interval", 1)
modparam("tm", "fr_timer", 1500) // default 30s
modparam("dispatcher", "ds_probing_threshold", 1)
modparam("dispatcher", "ds_inactive_threshold", 1)
```

limitations:

- ✗ Slow to react if timer timout value is too high
- ✗ Risk to run out of GW if timer timeout value is to low
- ✗ No memory of past problem is kept after recovering

Configuring load balancing with congestion detection

The reactivity when facing congestion can be tuned using the EWMA alpha, a larger alpha will result in an estimator with a **longer memory** and **faster reaction time**

```
modparam("dispatcher", "ds_ping_interval", 1)
modparam("dispatcher", "ds_ping_latency_stats", 1)
modparam("dispatcher", "ds_latency_estimator_alpha", 900)
modparam("dispatcher", "ds_latency_cc", 1) // use congestion control

    if ( is_method("INVITE") ) {
      if (!ds_select_dst("1", "11")) { // use relative weight based load distribution
            send_reply("404", "No destination");
            exit;
      }
    }
```

When facing congestion the **weight** of a gateway is lowered by 1 for every congestion ms.

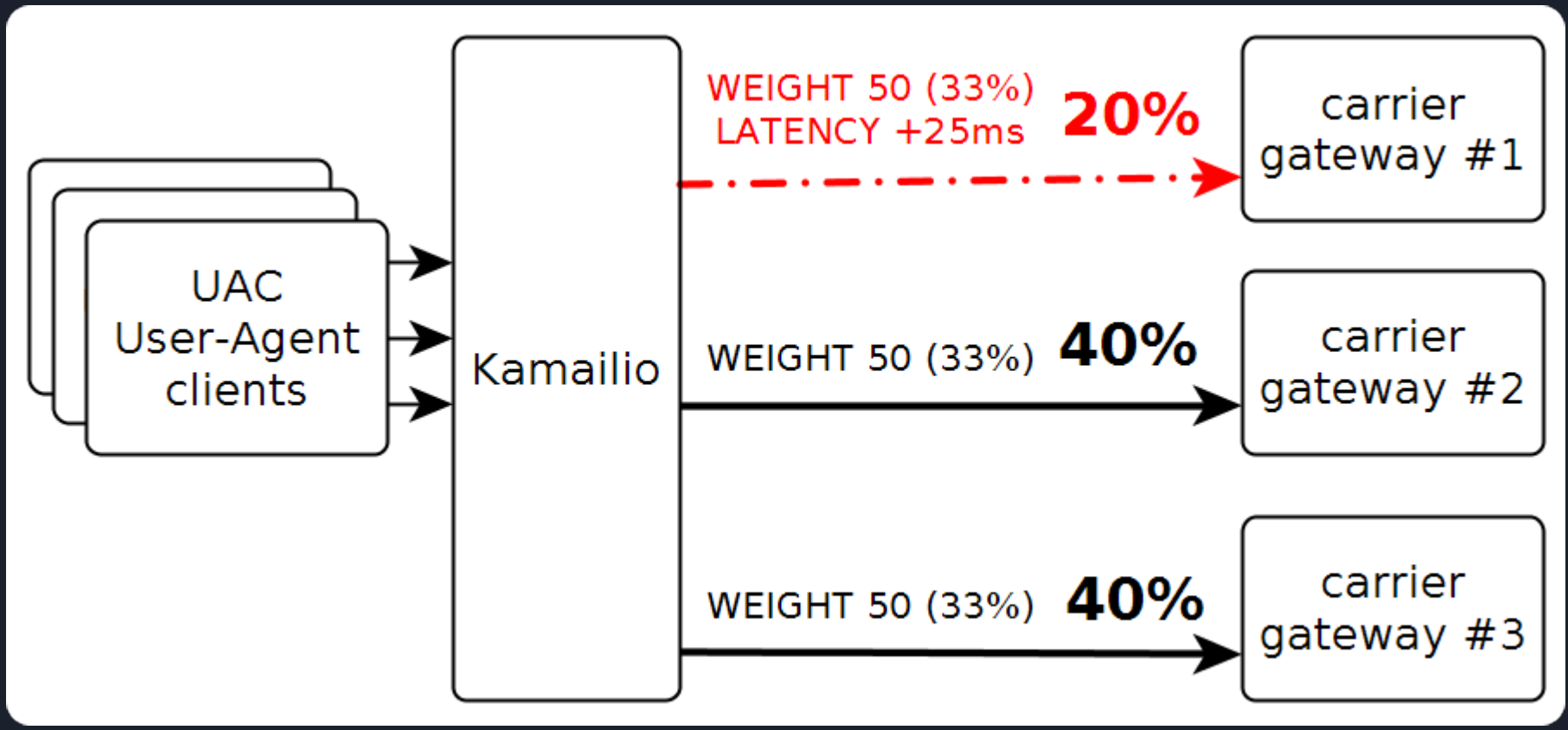
In this example the GW wills support up to 50ms of estimated congestion, therefore 50 is also the cut-off value. However, the GW will still be used if all the other GWs are also above their congestion threshold, in such case, load distribution will be based on the ratio of congestion_ms each GW is facing.

```
INSERT INTO "dispatcher" VALUES(1,1,'sip:1.1.1.1:5060',0,12,'weight=50;rweight=50','');
INSERT INTO "dispatcher" VALUES(2,1,'sip:2.2.2.2:5060',0,12,'weight=50;rweight=50','');
```

Expected behavior example :
One or more gateway facing congestion.

Estimated latency of 85ms while average latency is 60ms.

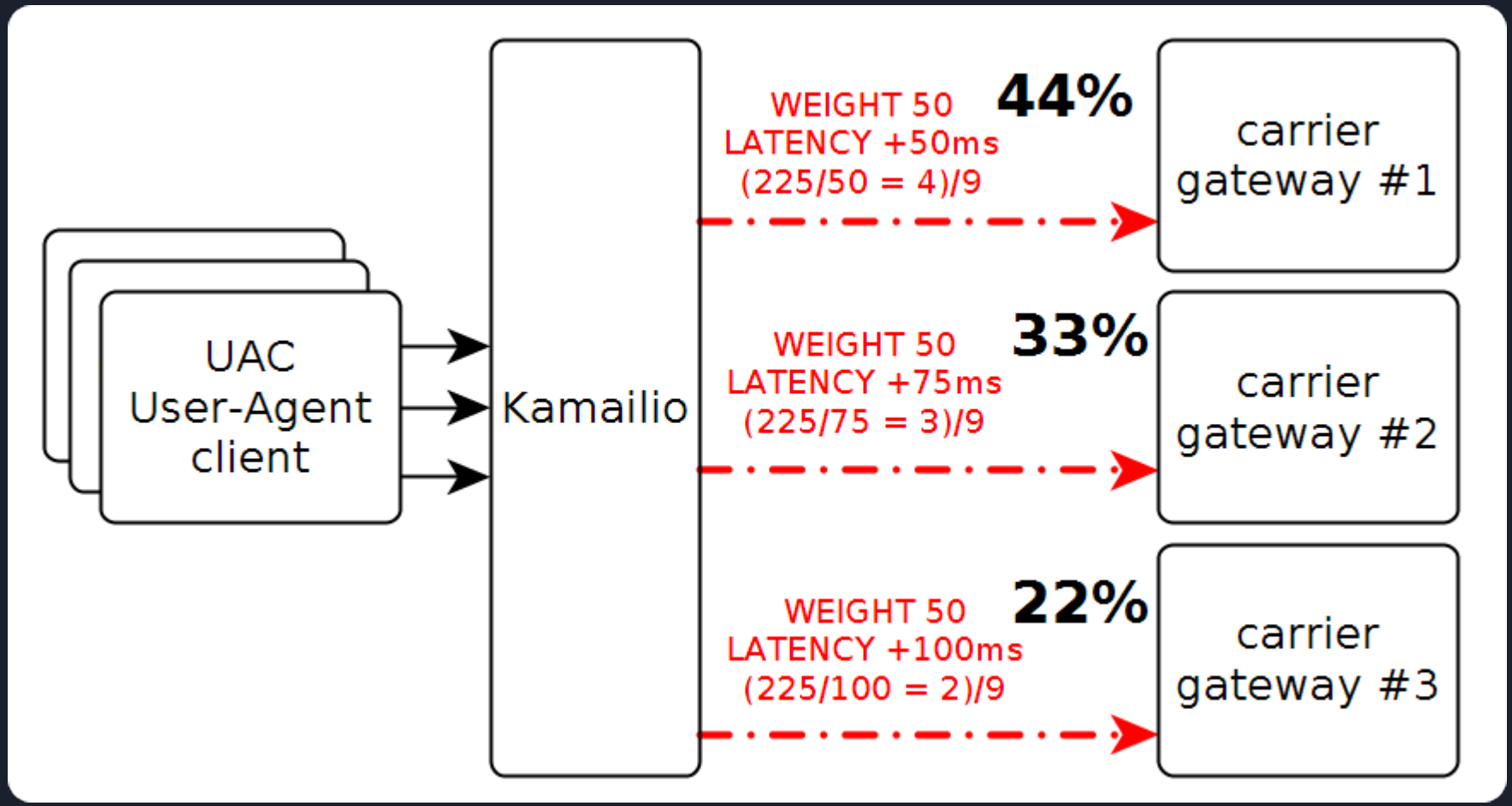
We remove 25 points of weight (one point of weight per ms of estimated congestion), we now have a ratio of 20% = 25/125



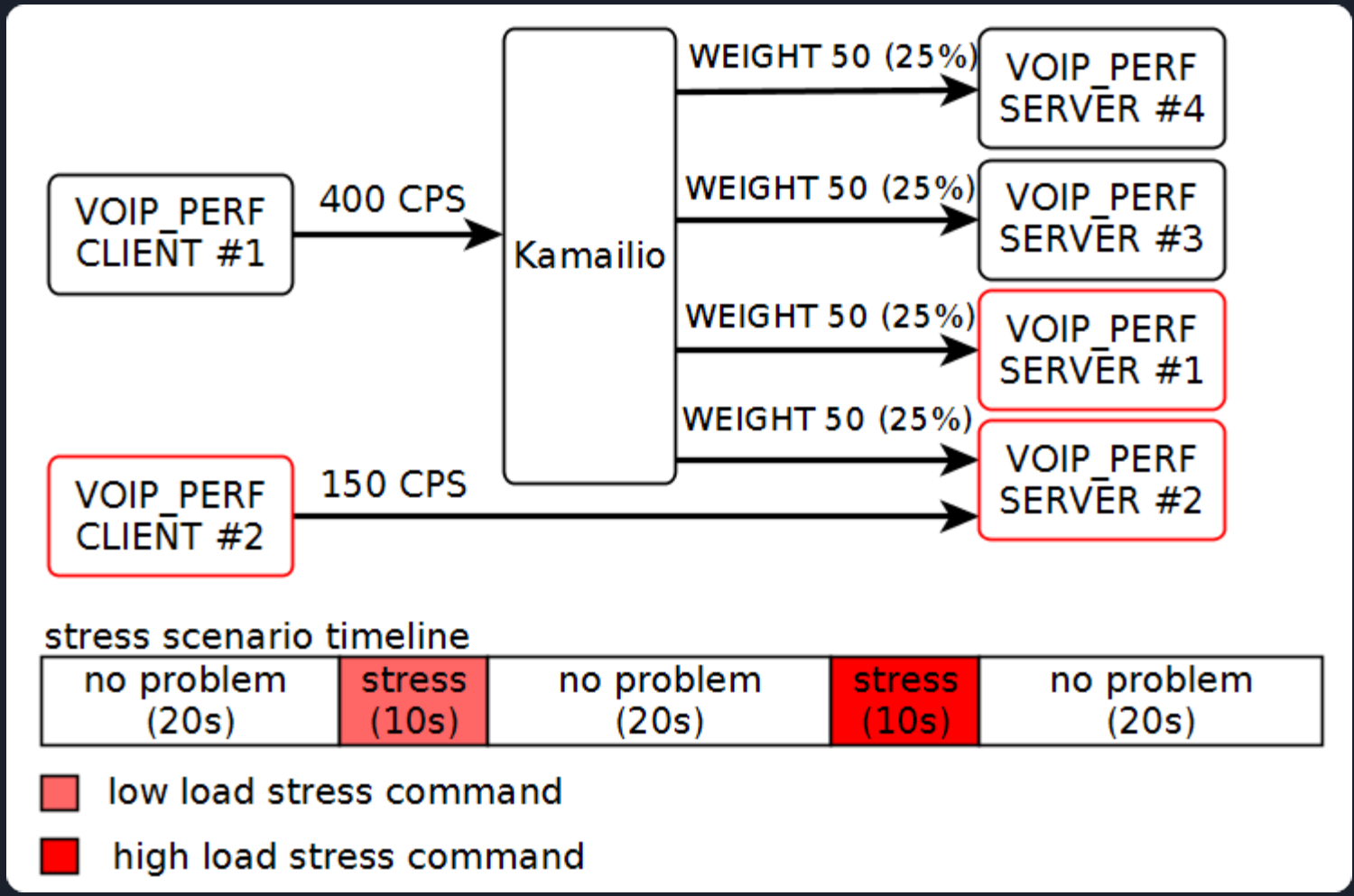
Expected behavior example :
All gateways congested

When the amount of estimated congestion is above the weight, the gateway is considered congested and will receive no traffic.

However when all the gateways are considered congested, the load distribution is done considering the ratio of congestion each gateway is facing.

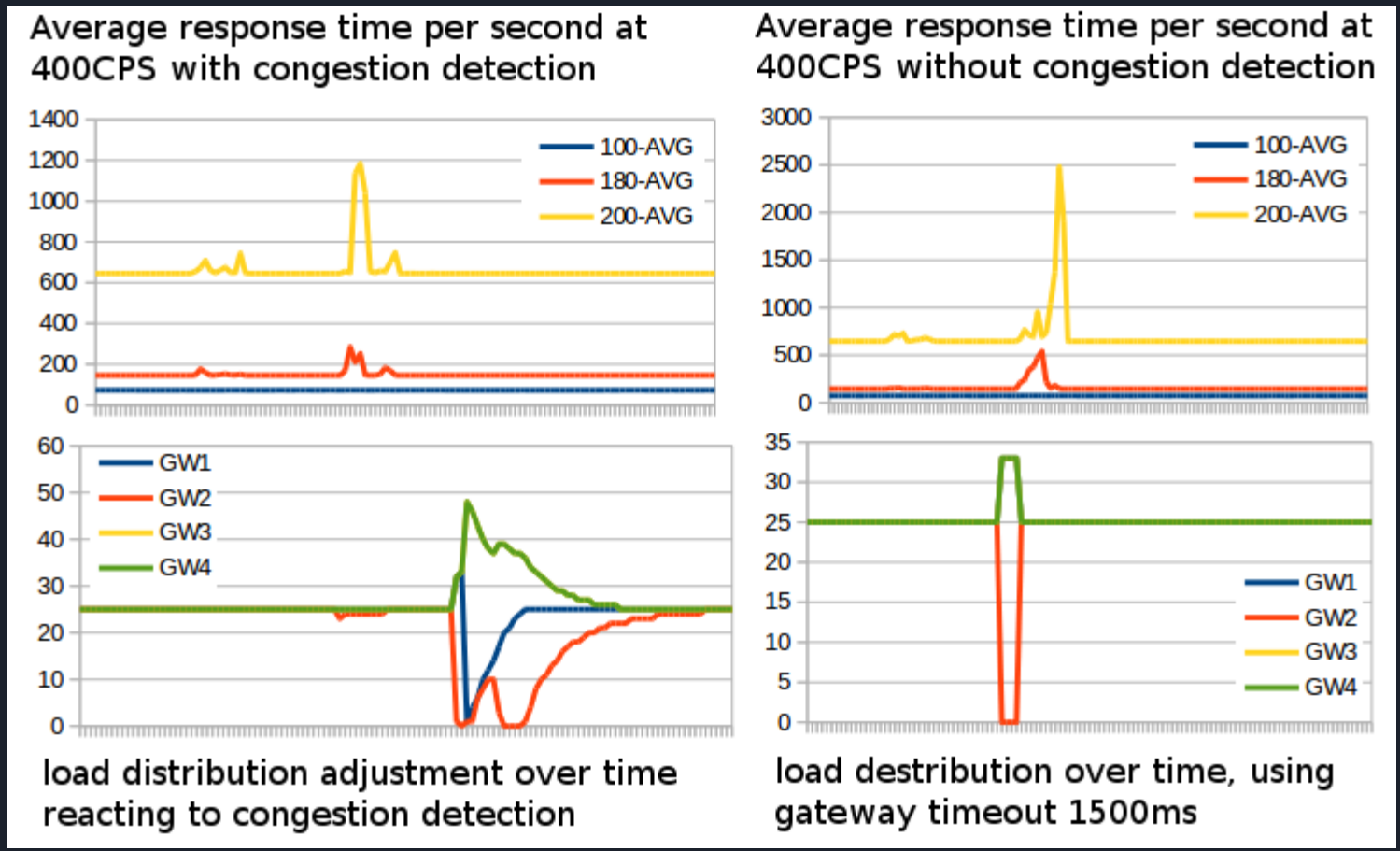


Lab tests and scenario



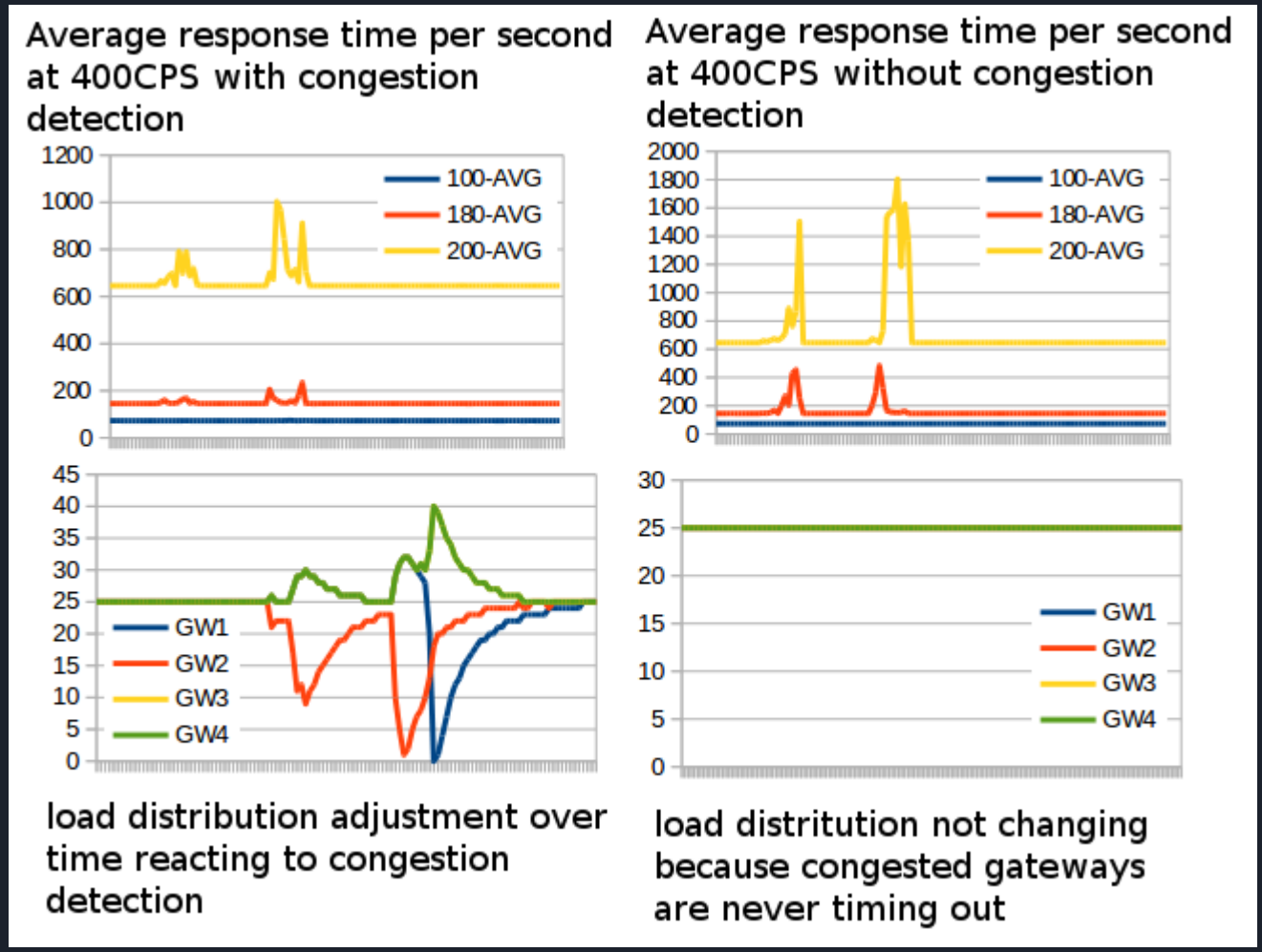
400 CPS responsiveness analysis with high SIP OPTIONS timeout value of 1500ms

modparam("tm", "fr_timer", 1500)



CONFIG	AVG 180	AVG 200	TIMEOUT	RECEIVED
no congestion control	157	688	0	50000
congestion control	149	660	0	50000

400 CPS responsiveness analysis with default (30s) SIP OPTIONS timeout value

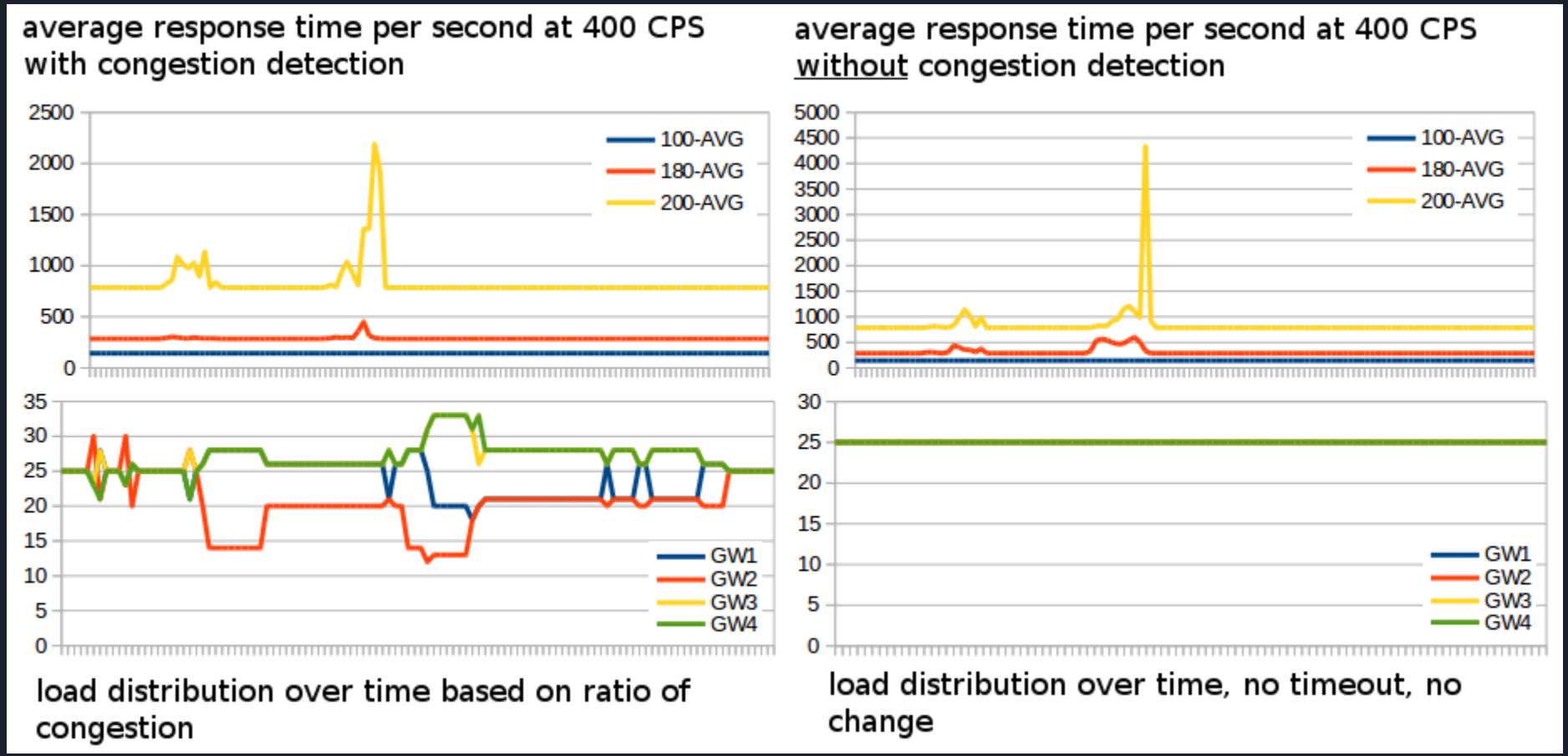


CONFIG	AVG 180	AVG 200	TIMEOUT	200 RECEIVED
no congestion control	159	709	216	49784
congestion control	147	662	0	50000

400 CPS, when all gateways congested

tc qdisc add dev bond0 root netem delay 70ms limit 125000

CONFIG	AVG 180	AVG 200	TIMEOUT	200 RECEIVED
no congestion control	307	838	0	50000
congestion control	288	833	0	50000



VOIP_PERF load test tool build on PJSIP

voip_perf command example :

```
./voip_perf -m INVITE -p $2 sip:+1206???????@14.75.64.225:5060 \
            --interval=1 \
            --count=50000 \
            --call-per-second=400 \
            --thread-count=1 \
            --window=100000 \
            --timeout 17200
```

voip_perf summary output :

```
Total 50000 INVITE calls sent in 124526 ms at rate of 401/sec
Total 50000 responses received in 125171 ms at rate of 399/sec:

Detailed responses received:
 - 200 responses:    50000     (OK)
                    ------
 TOTAL responses:    50000 (rate=399/sec)

Maximum outstanding job: 400
20:08:52.675    voip_perf.c  Peak memory size: 70MB
```

voip_perf latency file :

```
TIMESTAMP,METHOD,100-CNT,100-AVG,100-STD,100-MAX,180-CNT,180-AVG,180-STD,180-MAX,200-CNT,200-AVG,200-STD,200-MAX
1526286455,INVITE,400,142.45,0.50,143,310,285.07,0.41,286,48,785.62,0.64,787
1526286456,INVITE,400,142.50,0.50,143,489,285.05,0.35,287,430,786.66,24.10,1285
```

tmux voip_perf orchestration

```
#!/bin/bash
SESSION=dispatcher_perf_tests
tmux kill-session -t $SESSION
tmux -2 new-session -d -s $SESSION
tmux new-window -t $SESSION:1 -n 'Logs'
tmux split-window -h
# start voip-perf server
tmux select-pane -t 0
tmux send-keys "cd /git/voip_perf/ && ./voip_perf -p 5072 --trying --ringing --thread-count=1 -d 500" C-m
tmux split-window -v
# start voip-perf server
tmux select-pane -t 1
tmux send-keys "cd /git/voip_perf/ && ./voip_perf -p 5073 --trying --ringing --thread-count=1 -d 500" C-m
# Attach to session
tmux -2 attach-session -t $SESSION
```

Thank you for listening !

Looking forward working with you on Free Software

Tests configuration and results :
https://github.com/jchavanton/kam_load_balancing 

Voip Perf (based on PJ-SUA) :
https://github.com/jchavanton/voip_perf 

Thanks to Flowroute for supporting my trip to Kamailio World 2018 and
Other Free software events !

Thanks to :
Amy Meyers @ Flowroute (help with Algorithm design, testing and review)